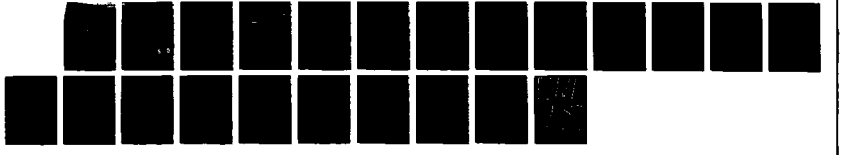
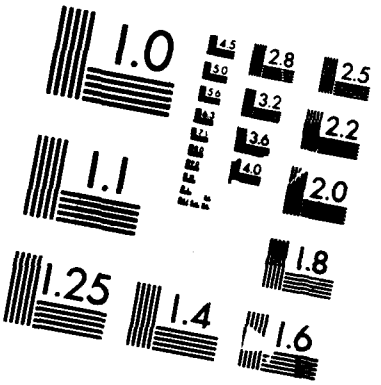
AD-A183 415 THE LOOM KNOWLEDGE REPRESENTATION LANGUAGE(U) 1/1
UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY
INFORMATION SCIENCES INST R MACGREGOR ET AL. MAY 87
UNCLASSIFIED ISI/RS-87-188 MDA903-81-C-0335 F/G 12/9 NL





ISI Reprint Series
ISI/RS-87-188
May 1987

13

University of Southern California

Robert Mac Gregor
Raymond Bates

The Loom Knowledge Representation Language

Reprinted from the
Proceedings of the Knowledge-Based Systems Workshop,
held in St. Louis, Missouri, on April 21-23, 1987.

INFORMATION SCIENCES INSTITUTE

213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT This document is approved for public release, distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) ISI/RS-87-188	5. MONITORING ORGANIZATION REPORT NUMBER(S) ----------
6a NAME OF PERFORMING ORGANIZATION USC/Information Sciences Institute 6b OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION ----------
6c ADDRESS (City, State, and ZIP Code) 4676 Admiralty Way Marina del Rey, CA 90292	7b. ADDRESS (City, State, and ZIP Code) ----------
8a NAME OF FUNDING / SPONSORING ORGANIZATION DARPA 8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-81-C-0335
8c ADDRESS (City, State, and ZIP Code) Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: ---- ---------- PROJECT NO: ---------- TASK NO.: ---------- WORK UNIT ACCESSION NO.: ----------

11 TITLE (Include Security Classification)
The Loom Knowledge Representation Language [Unclassified]

12 PERSONAL AUTHOR(S)
Mac Gregor, Robert; Bates, Raymond

13a TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Research Report	FROM _____ TO _____	1987, May	22

16 SUPPLEMENTARY NOTATION
Reprinted from the *Proceedings of the Knowledge-Based Systems Workshop*, held in St. Louis, Missouri, on April 21-23, 1987.

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP
09	02	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
classification-based reasoning, KL-ONE, knowledge representation languages, structured- inheritance networks

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The lengthening lifetimes of intelligent systems, and the desire to share or re-use knowledge bases, has created within the AI community the need for application-independent knowledge representation systems. The Loom system being developed at ISI represents the latest in a series of "classification-based" knowledge representation systems developed to meet this need. In Loom, the traditional single-classifier architecture is replaced by one containing a collection of classifiers which exhibit increasingly powerful inference capabilities. This paper describes the knowledge representation language developed for the Loom system.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Sheila Coyazo Victor Brown	22b TELEPHONE (Include Area Code) 213-822-1511 22c OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

ISI Reprint Series
ISI/RS-87-188
May 1987

University of Southern California

Robert Mac Gregor
Raymond Bates

The Loom Knowledge Representation Language

Reprinted from the *Proceedings of the Knowledge-Based Systems Workshop,* held in St. Louis, Missouri, on April 21-23, 1987.

INFORMATION SCIENCES INSTITUTE

213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract MDA 903 81 C 0335. Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion of DARPA, the U.S. Government, or any person or agency connected with them.

ISI Reprint Series

This report is one in a series of reprints of articles and papers written by ISI research staff and published in professional journals and conference proceedings. For a complete list of ISI reports, write to

Document Distribution
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
USA

The Loom Knowledge Representation Language

Robert Mac Gregor
Raymond Bates

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Abstract

The lengthening lifetimes of intelligent systems, and the desire to share or re-use knowledge bases, has created within the AI community the need for application-independent knowledge representation systems. The Loom system being developed at ISI represents the latest in a series of "classification-based" knowledge representation systems developed to meet this need.[1] In Loom, the traditional single-classifier architecture is replaced by one containing a collection of classifiers which exhibit increasingly powerful inference capabilities. This paper describes the knowledge representation language developed for the Loom system.

1. Introduction

Loom[2] represents a recent entry into the KL-ONE [Brachman and Schmolze 85] family of knowledge representation systems. Loom directly succeeds the NIKL system [Schmolze and Lipkis 83, Moser 83] developed jointly by ISI and BBN. During NIKL's lifetime, the NIKL user community produced a rather extensive list of extensions that they wished to see in future versions of NIKL [Kaczmarek 86]. Loom's designers determined that these needs could best be achieved by redesigning and reimplementing NIKL. The result is a more flexible architecture which preserves the strengths of the original NIKL while admitting some new and powerful forms of reasoning.

Loom's architecture strongly reflects the view that the variety of inferences provided by a comprehensive knowledge representation system can best be performed by a well-integrated collection of specialized reasoning components, rather than by a single, general-purpose reasoner. KL-ONE-style systems (e.g., KL-ONE, KL-TWO [Vilain 85], KRYPTON [Brachman, Fikes, and Levesque 83], and BACK [von Luck 87]) have traditionally divided their knowledge space into two partitions, called the "Terminological Box" and the "Assertional Box", and have utilized two distinct reasoners (terminological and assertional) to carry out their inferences. Loom's principle architectural contribution is to introduce two additional partitions (the "Universal Box" and the "Default Box"), each having its own associated reasoning component.

Complementing this increase in the number of domain-independent reasoners embedded in the system architecture is a growing library of domain-specific, "narrow-coverage" reasoners. Currently these include facilities for computing or reasoning about transitive relations, sets, intervals, and some elementary forms of numeric reasoning. These reasoners can be invoked independently, or called by the broad-coverage reasoners.

The trick in integrating this collection of reasoners is to develop a language for expressing knowledge which emphasizes the overall coherence and uniformity of the knowledge structures. Loom accomplishes this goal by building on the "concept-centered" view of knowledge employed in KL-ONE (and NIKL). Accordingly, all universal and default knowledge is attached to specific

[1]This research is supported by the Defense Advanced Research Projects Agency under Contract MDA903-81-C-0335. Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion of DARPA, the U.S. Government, or any person or agency connected with them.

[2]Loom: "A frame for interlacing sets of threads or yarns to form a cloth." Webster's

concepts. In a similar vein, sets, intervals, and relations (including transitive and composite relations) are all realized as specialized forms of concepts -- their definitions share a uniform syntax, and each of them has its own sublattice within the concept taxonomy.

This paper introduces the syntax and semantics of that portion of the Loom knowledge representation language which represents meta-level knowledge. We include discussions on some of the types of inference which can be performed by the Loom system. We begin by defining the four broad types of knowledge managed by the Loom system, and then discuss each of the "Boxes" devoted to representing meta-level knowledge. The appendices include the knowledge bases used to illustrate examples of Loom syntax. A longer version of this paper [Mac Gregor 87] contains a complete definition of the Loom system.

2. Boxes

In order to accurately define concepts and relations in Loom, it is necessary to have an understanding of how Loom treats various "kinds" of knowledge within the system. Loom partitions its knowledge space into four "Boxes", called the Terminological, Universal, Default, and Assertional Boxes. This section presents a brief characterization of each of these four kinds of knowledge. Later sections will present specifics on the expressive features available with each of the Loom boxes.

Definitions within the "Terminological Box" (TBox) serve to define the "terms" in our knowledge representation scheme ([Brachman, Fikes, and Levesque 83] contains a good discussion of what kind of knowledge is considered to be "terminological"). A TBox definition yields a set of necessary and sufficient conditions for recognizing an instance of some concept. Within Loom, the organization (classification) of concepts is based strictly on the terminological knowledge available to the system.

The "Universal" Box (UBox) widens the scope of things we can say about (generic) concepts to include certain forms of knowledge about the "real world". In the UBox we can attach necessary conditions to a concept definition. For example, we can state that "live-persons necessarily have heads", i.e.,

$\forall x[Live\text{-}Person(x) \rightarrow \exists y\, head(x, y)]$.

In the UBox we can also state conditions which are sufficient, but not necessary to recognize an instance of a concept. For example, we can say that "all featherless bipeds are human", i.e.,

$\forall x[Featherless\text{-}Biped(x) \rightarrow Human(x)]$.

A second, more powerful classifier is associated with the UBox. The UBox classifier makes its inferences (classifications) on the basis of combined TBox and UBox knowledge.

The "Default" Box is the proper location for representing "assumptions" or "default knowledge". For example, in it we can state such things as default values: "If nothing has been asserted about the color of some elephant x, make the assumption 'color(x Grey)'." We can also state some limited forms of closed-world assumptions:[3] "If some paper P has K authors, assume that it has *only* K authors."

The knowledge represented in the Default Box is used to make some very limited types of inferences during the process of realization. A full-blown use of default knowledge would seem to require the inclusion of a non-monotonic reasoning capability into Loom. This is beyond the scope of our current effort.

The Assertional Box (ABox) is the repository for assertions about individuals. For example, we might

[3] By default, the ABox assumes open-world semantics.

place in the ABox the assertion that Clyde is a white elephant by making the assertions:

```
(assert (Elephant Clyde) (color Clyde White)).
```

The effect of these assertions is to create an instance in the ABox of the concept `Elephant` (unless `Clyde` already exists in the ABox) and to assign to the `color` role of the object `Clyde` the value `White`.

Loom has extended NIKL's terminological language CNIKL [Robins 86] to include expressions of universal and default knowledge. We believe that it is beneficial to associate each fragment of universal or default knowledge with a particular concept; thus, we have chosen to extend the syntax of the original `defconcept` (and `defrelation`) primitives, rather than to add new (top-level) constructs to the terminological language. The Engines and Cars knowledge base in Figure A-1 illustrates some Loom concept declarations. The original CNIKL definition of a concept serves as its definitional component. An "axioms" clause states universal knowledge about a concept, while a "defaults" clause states default knowledge.

Engineering Note:

Our introduction of a new type of reasoner (the TBox classifier) puts us in line with what we see as a long-range trend towards knowledge representation architectures which will employ increasing numbers of specialized reasoners. As the number of reasoners within a single system increases, it will become increasingly important that some organizing principle is available to integrate these various reasoners. Our decision to organize all universal and default knowledge within the context of particular concepts illustrates a belief that the "concept-oriented" (a.k.a. "frame-oriented") approach will prove to be a successful organizing principle for wider and wider classes of knowledge. Such an approach may be contrasted with that of the current generation of rule-based systems (including hybrid frame- and rule-based systems); in those systems, knowledge which we have classed as universal or default knowledge (other than "default values") tends to be dumped unceremoniously into a "rule base", i.e., such systems provide no formal scheme for structuring that knowledge.

3. Basic Terminology

Here we take time-out to formalize some of our terms.

By a *concept* we mean an "intentional description" of something. The most general instance of a concept is called "`Thing`". A *relation* is a concept which defines a set of k-tuples, with k being fixed for each individual relation. By convention, the the term "concept" is often used to refer to (the more specialized notion of) a *unary relation*. Thus, the `defconcept` form defines a unary relation.

A binary-relation for which the roles domain and range have been assigned will be called a *mapping*. By convention, the term "relation" may be used in place of the word "mapping", and the form `defrelation` is used to define a mapping. The most general instance of a mapping is called "`maps-to`".[4] The Loom implementation is intended to accommodate relations of order greater than two, but a complete syntax for defining higher-order relations has not yet been worked out. A relation which has been reified (equated with a unary concept of the same name) is termed a *relationship*.

The domain of a mapping is not considered to be a part of its (TBox) definition. The association of a mapping with a particular (domain) concept, other than the concept THING, induces a sub-relation we call a *role*.[5] A *role restriction* which associates a mapping M with a concept C defines a role R_{CM} such that R_{CM} is a subset of M, and has domain C. A *value restriction* is a role

[4] "maps-to" corresponds to the NIKL relation "MostGeneralRole"

[5] Roles are seen as virtual objects in Loom, i.e. there are no structures in the system which can be identified as roles

restriction which restricts the range of R_{CM}, while a *number restriction* is a role restriction which places bounds on the number of role fillers of R_{CM} that can be associated with a single instance of C. A composition of mappings M_1, ..., M_k such that the domain of M_1 is restricted to a particular concept (other than THING) is called a *role chain*.

Loom distinguishes between "primitive" and "defined" concepts (and relations). A concepts is *primitive* if no complete definition can be given for it (see [Vilain 84, p. 549 or , Brachman and Schmolze 85]); otherwise it is *defined*. Concepts and relations are organized into a taxonomy based on a partial-ordering relation called "specializes". A concept C_1 *specializes* a concept C_2 if and only if membership in C_1 entails membership in C_2, i.e. iff

$$\models \forall x[C_1(x) \rightarrow C_2(x)].$$

An instance of a specializes relation between two concepts may be declared explicitly in a concept definition, or it may be deduced by the classifier.

A *value* is an object which corresponds to a logical constant in a knowledge base, and is typically left undefined in a knowledge base. The numbers 1, 3, and 8.2, and the sexes Male and Female are examples of values. A concept which is defined by enumerating its instances is called a *set*. Currently, all of the sets we define in the TBox are sets of values. Number and Sex are examples of sets. A (denumerable) set for which precedessor and successor relations exist is termed an *interval*, e.g., Integer and Days-of-the-Week are intervals.

To *classify* a concept means to link it into the specialization lattice so that (i) it is below all concepts which it specializes, and (ii) it is above all concepts which specialize it. The *most specific generalization* (MSG) of a concept is the set of those concepts which are/would become its direct ancestors (parents) if it were classified.

To *recognize* an ABox object/instance x means to compute the set of concepts $\{C_i\}$ such that for each C_i, x is an instance of C_i, and x is not an instance of any descendant of C_i. The set $\{C_i\}$ is referred to as the MSG of x. In an informal discussion we may use the term "classification" to refer to either the classification or the recognition process.

4. The TBox

In this section we present the syntax and semantics of TBox definitions for (unary) concepts, (mapping) relations, sets and intervals. Occasionally within this discussion we will pause to point out some of the deductions which the Loom classifier will (or will not) be able to make. These comments are intended to foster an appreciation for what kinds of inference one can expect from a classifier. Next comes a brief discussion outlining our reasons for prohibiting cyclically-defined concepts, and we conclude with a presentation of three additional restrictions which Loom imposes on TBox definitions.

4.1. Defconcept and Defrelation

A formal semantics for the term-forming operations `defconcept` and `defrelation` appears as Appendix B. The simple definitional constructs listed in the figure can be combined within a concept or relation definition to form compound definitions. The semantics for such a compound definition are defined as the logical conjunction of the individual lambda definitions.

For example, referring to the Engines and Cars KB in Figure A-1, suppose we declare a new concept

```
(defconcept ( specializes Engine)
      ( :restriction cylinders ( min 4) ( max 6))
      ( :restriction fuel ( vr Gasoline)))
```

This concept means "an engine fueled by gasoline which has between 4 and 6 cylinders." The TBox classifier will discover that this concept specializes the concept labeled `Internal-Combustion-Engine`.

The Familial Relations KB in Figure A-4 illustrates how **defrelation** constraints can be combined to form terms for the relations **parent**, **father**, **grandfather**, etc. The classifier will determine, among other things, that **grandfather** specializes **grandparent** and that **parent** and **grandparent** specialize ancestors. A few short-hand notations are provided in addition to the operators illustrated in Figure A-4. The following pairs of forms are equivalent:

The forms

```
(:restriction M (:number k))                    and
(:restriction M (:min k) (:max k)),
```

the forms

```
(:restriction (:vrdiff M C) ...)        and
(:restriction
   (defrelation (:specializes M) (:range C)) ...),
```

the forms

```
(:restriction (:closure-of M) ...)      and
(:restriction (defrelation (:closure-of M)) ...).
```

Loom's *constraint* clause extends the CNIKL construct referred to as a "role-constraint" or "role-value-map" by (1) allowing for other operators than just set-equality and set-containment, and (2) allowing a value to take the place of a role-chain. The argument "CP" in the clause

```
(:constraint CP (   ) (   ))
```

must name a relation which falls in the sublattice rooted at the relation **Compute-Relation**. Figure A-2 illustrates some compute relations. The operators for computing set-equality, set-inequality, and set-containment are other examples of compute relations.

Again referring to the Engines KB, let us declare two new concepts:

```
(defconcept Big-Engine
     (:constraint greater-than (horse-power) 120))
(defconcept Very-Big-Engine
     (:constraint greater-than (horse-power) 200))
```

We plan to upgrade the Loom classifier so that it will be able to deduce that **Very-Big-Engine** specializes **Big-Engine**. The analysis will necessitate recognizing the truth of **(greater-than 200 120)**, and will involve reasoning about the transitivity of the **greater-than** relation. During a 1986 NIKL users workshop [Moore 86], Ron Brachman discussed the possibility of extending a NIKL-like system to include a couple of new "boxes" in addition to the traditional TBox and ABox. One of those boxes he termed a "Mathematics Box", which would be a specialized reasoner with the ability to derive mathematical inferences in conjunction with the TBox reasoner. The numerical reasoning facility just hinted at represents an embryonic step in the direction of developing a full-fledged mathematics box.

We will conclude this section will an example containing definitions for which Loom cannot deduce the implied subsumption relations. Referring to the Familial Relations KB again, consider the following definitions of a concept named "Only-Child":

```
(defconcept Only-Child-1
        (:constraint equals self (parent child)))
(defconcept Only-Child-2
        (:restriction siblings (:max 0))).
```

The current Loom classifier cannot deduce that the concepts **Only-Child-1** and **Only-Child-2** are equivalent. The NIKL classifier is similarly unable to deduce this equivalence relation (when applied to CNIKL analogues of the above definitions). Our current development philosophy is that we are committed to developing a system which makes inferences which are sound, but not necessarily complete. One of the philosophical goals of the Loom system is to investigate empirically where the boundaries should be on the expressive power of a TBox. Once those bounds have been more-or-less established, it may be appropriate to revive the goal of developing a reasoner which is as complete as we can make it.

4.2. Defset and Definterval

This section describes the operators **defset** and **definterval**, which can be employed to define sets and intervals, and also to define concepts corresponding to the values enumerated in those sets/intervals. Our ex-

amples will reference the Sets and Intervals KB in Figure A-3.

In many cases, there is a tight coupling between values in a set or interval which represent "qualities" (e.g., the sex `Male` or the color `Red`) and concepts such as `Male-Animal` or `Red-Thing` which are defined by having one of their attributes restricted to the corresponding value: Definitions for `Male-Animal` and `Red-Thing` might be

```
(defconcept Male-Animal (:specializes Animal)
      (:restriction sex (:vr Male)))
(defconcept Red-Thing
      (:specializes Monochrome-Thing)
      (:restriction color (:vr Red))).
```

Thus, we have

$$\forall x[(Animal(x) \wedge sex(x, Male)) \leftrightarrow Male\text{-}Animal(x)].$$
$$\forall x[(Monochrome\text{-}Thing(x) \wedge color(x, Male)) \leftrightarrow Red\text{-}Thing(x)].$$

The Loom syntax for sets and intervals includes an optional "partitions" clause which produces the set of definitions needed to characterize this behavior.

The declaration

```
(defset Sex (:values Male Female))
```

defines a set `Sex` and the values `Male` and `Female`. To introduce the concepts `Male-Animal` and `Female-Animal`, we can augment our definition with the clause `(partitions Animal)` (Figure A-3 illustrates the complete definition). This larger declaration implicitly declares the following expressions:

```
(defrelation Sex :primitive
  ( axioms ( domain Animal) ( range Sex)))
(defconcept Male-Animal (:specializes Animal)
  ( restriction Sex Male))
(defconcept Female-Animal ( specializes Animal)
  ( restriction Sex Female))
```

In addition, the declaration for the concept `Animal` is augmented by a clause which indicates that `Male-Animal` and `Female-Animal` form a disjoint covering of Animal.

We next turn our attention to the interval `Naval-Rank` defined in Figure A-3. The declaration of `Naval-Rank` implies the definition of a relation `Naval-Rank`, and also implies the declaration of the concepts `Seaman-Recruit`, `Seaman-Apprentice`, ... , `Admiral`. [6] The implied declaration for `Admiral` is

```
(defconcept Admiral (:specializes Naval-Person)
   (:restriction Naval-Rank Admiral)).
```
[7]

Because Naval-Rank is specified as an interval, rather that as a set, the relations "successor" and "predecessor" are defined for its instances. Their definition corresponds to the order of values in the "values" clause. For example, `(successor Commander Captain)` is true. The successor and predecessor relations may appear within the role chains of a constraint clause. A square-bracket notation can be employed to define a (contiguous) subset of an interval. This is illustrated in the definition of the set `Naval-Officer-Rank`, and in the definitions below that for `Natural-Number`, `Positive-Integer`, and `Non-Negative-Integer`. The semantics of subsumption for intervals is the same as that for sets. For example, the interval defined by

```
(definterval (:specializes Integer)
      (:values 3 7 5))
```

specializes the interval defined as

```
(definterval (:specializes Integer)
      (:values [2..9])).
```

4.3. How to Avoid Cycles

A concept (or relation) definition *depends on* another definition if it references the other concept by name within its definition. If these depends-on links form a cycle, then we say that the definitions involved are *cyclic*. The designers of the NIKL system expressly permitted cyclic definitions. However, the semantics associated with cyclic CNIKL definitions was never fully worked out, and the behavior of the NIKL classifier when it encountered cycles was far from satisfactory. Loom has taken an opposite position -- cyclic definitions are illegal in Loom.

---

[6] In the declaration of "Naval-Rank", the clause "(suffix Nil)" prevented the suffix "-Naval-Person" from being appended to each new concept

[7] Observe that the concepts "admiral as person" and "admiral as naval-rank" have the same name. Loom will automatically add suffixes "-1" and "-2" to distinguish between them

A primary motivation for allowing cycles was to avoid placing a restriction on what concepts could appear within a value restriction clause. Consider the following definition of Human:

```
(defconcept Human :primitive (:specializes Mammal)
      (:restriction parents (:vr Human)))
```

The value restriction (:vr Human) allows the system to infer "If an individual is Human, then so are its parents, and their parents, and so on." Because that value restriction is self-referential (defining a cycle of length one), it is not permitted in Loom. However, Loom does allow an equivalent restriction to be expressed as an axiom in the UBox:

```
(defconcept Human :primitive (:specializes Mammal)
    (:axioms
        (:restriction parents (:vr Human))))
```

Thus, we retain in Loom the ability to make statements such as, "the parents of humans are also human"; we just don't allow them to be included as a part of the (terminological) definition of a concept.

5. The UBox

The knowledge which we place in the Universal box augments individual TBox definitions with what we call universal or contingent knowledge. The expressive power of the Loom language increases significantly when the definitional language is extended to include expressions of universal knowledge. This combined language admits a correspondingly larger class of inferences.

This section will first define the different types of knowledge which we class as "universal". Next, we introduce the notion of a "stable" classifier, which serves to sharpen the definitional boundary between terminological and universal knowledge. Finally, we will present the representational model and classification algorithm adopted by the Loom architecture to handle universal knowledge.

5.1. Types of Universal Knowledge

In anticipation of our later discussion on how Loom represents universal knowledge, we will group our universal knowledge into four categories. Referring to universal knowledge that is attached to a concept "P", the categories are:

1. Contingent restrictions and constraints -- these are restrictions or constraints which necessarily apply to an instance "x" if P(x) holds. These are often called "necessary conditions";

2. Implications -- these are statements of the form "P implies Q" (where Q is a concept which does not subsume P). Often called "sufficient conditions";

3. Equivalences -- these are statements of the form "P if and only if Q". Often called "necessary and sufficient conditions";

4. Other non-definitional knowledge about concepts and relations. Currently this knowledge consists of covering relations, disjointness relations, marking concepts as "individual", and domain and range constraints on mappings.

5.1.1. Contingent Restrictions and Constraints

The "axioms" clause of a concept or relation definition states universal knowledge which applies to that concept or relation. The Engines and Cars KB of Figure A-1 illustrates several such clauses. The next few examples will be drawn from that KB.

The clause

```
(:axioms (:res (:vrdiff has-component Engine)
               (:number 1)))
```

which appears in the definition of Car is an example of a "contingent restriction". The meaning of the clause is

$$\forall x[Car(x) \rightarrow \exists\ \textit{exactly one}\ y\ (\textit{has-component}(x, y) \wedge \textit{Engine}(y))].$$

This is sometimes referred to as a "necessary condition" because it translates as "it is *necessarily* the case that a car has exactly one engine." In general, the meaning of a restriction (or constraint) appearing within an "axioms" clause of a defconcept form defining a concept C is, "this restriction (constraint) applies to all objects which are instances of C".

5.1.2. Implications and Equivalence Relations

The clause (:axioms (:implies Car)) which appears within the defconcept form which defines Battery-Powered-Vehicle is an example of an *implication*. Its meaning is

$\forall x[Battery\text{-}Powered\text{-}Vehicle(x) \rightarrow Car(x)]$.

This form of knowledge is sometimes called a "sufficient condition" because it can be translated as "to determine is x is an Car, it is *sufficient* to determine that x is a Battery-Powered-Vehicle."

It is important to distinguish the difference in semantics between an implication (an "implies" relation) and a "specializes" relation. While the logical form associated with each of them is identical, the semantics of the specializes relation is significantly stronger. The statement "B specializes A" says not only that (1) B *implies* A, but also that (2) B's (TBox) *definition* includes the definition of A, and (3) B *inherits* the (UBox) properties of A.

A two-way implication established between a pair of concepts defines an *equivalence* relation. More generally, any cycle of implications through a set of concepts establishes an equivalence relation between each pair of concepts in that set. Suppose a set of concepts $\{C_i\}$ have been defined such that they are pairwise-equivalent. While the TBox sees the C_i as distinct concepts, the UBox view of this knowledge sees a single concept C_U which combines all of the knowledge declared in each of the C_i *(this is described in more detail in section* 5.3.). This means that universal knowledge (other than the "implies" relations) can be distributed in any number of ways among the C_i's, and the semantics will always be the same.

The *preferred* way to model a set of equivalent concepts $\{C_i\}$ is to explicitly declare an additional concept C which specializes each of the C_i, and which contains all of the universal knowledge associated with the C_i, except for a clause (axioms (:implies C)) which appears in each of the C_i definitions. Our definition of the concepts Diesel-Oil-Engine, Thing-With-Glow-Plugs, Very-High-Compression-Engine, and Diesel-Engine in Figure A-1 illustrates this type of modeling.

5.1.3. Coverings and Disjointness Classes

A *covering* for a concept "A" is a set of concepts whose union contains A. Loom syntax requires that the concepts within such a covering specialize A, so that the union of the covering concepts *equals* A. The meaning of the clause (:axioms (:covering B C)) within a defconcept for A is

$\forall x[A(x) \rightarrow (B(x) \vee C(x))]$.

Declarations of unary coverings (coverings containing a single concept) are illegal in Loom because they are logically equivalent to "implies" relations, and hence are redundant.

A *disjointness class* is a set of concepts which are declared to be mutually disjoint. A disjointness class is always defined with respect to a concept which subsumes the members of the class. The meaning of the clause

(:axioms (:disjoint B C))

within a defconcept for A is

$\forall x[B(x) \leftrightarrow \neg C(x)]$.

A *disjoint-covering* of a concept A enumerates a set of concepts which partition A, i.e., it is interpreted as the logical conjunction of a covering declaration and a disjointness declaration.

The Numeric-Comparison KB in Figure A-2 illustrates some declarations of coverings and disjoint-coverings. The covering defined for the relation numeric-comparison declares that the relations greater-or-equal and less-or-equal cover numeric-comparison. The disjoint-covering declaration for greater-or-equal states both that greater-or-equal is covered by greater-than and equal, and that the relations greater-than and equal are disjoint. Loom

provides functions for asking questions about (declared or derived) disjointness and covering relations, such as "Are concepts A and B disjoint?", "Do concepts A, B, and C cover concept D?", or "List all coverings for concept D".

Loom requires that the concepts or relations appearing in a covering, disjointness class, or disjoint-covering must all be *primitive*. The philosophical justification for this restriction is that if one or more of the members of the covering and/or disjointness class are not primitive then either (i) the covering and/or disjointness relations could have been logically inferred on the basis of other knowledge or (ii) such relation(s) could be derived. In the former case, the covering and/disjointness declaration is redundant, and should be dropped. In the latter case, there must have been something left unstated about the non-primitive concepts, which suggests that they are in fact primitive.

The implementors of the NIKL system encountered a practical reason for requiring members of a disjointness class to be primitive. That restriction prevented an anomaly which arose in a situation in which so-called "incoherent" concepts were being classified. The possibility of a similar anomaly arising in conjunction with covering declarations has not yet been explored.

We are considering omitting the `disjoint` clause altogether from the Loom language, owing to the observation that we have not yet encountered the use of a disjointness declaration in a context where an obvious covering relation did not also exist, i.e., where `disjoint-covering` could not have been substituted. Our syntactic requirement that a disjointness class be defined relative to a particular concept anticipates this future restriction.

5.1.4. Domain and Range Restrictions

"Domain" and "range" clauses which appear within an "axioms" clause state necessary conditions about a relation. The declaration

```
(defrelation M ...
   (:axioms (:domain A) (:range B)))
```

makes the universal statement

$$\forall xy[M(x,y) \rightarrow A(x) \wedge B(y)].$$

Knowledge about domain and range constraints is referenced during the "model-building" phase, when the initial definitions of concepts and relations are being refined and checked for coherence. In this context, these constraints function as "integrity constraints".

5.1.5. Individual Concepts

Marking a concept as "individual" means that its extension has cardinality at most one. We have identified some inferences that can be made on the basis of individual markings on concepts, but none of these inferences are particularly useful. Thus, this feature currently serves only as a place-holder, awaiting a user who will conceive of a use for it.

The presence of the "individual" marking is a part of Loom's NIKL heritage. Because most applications of NIKL operated without an ABox -- individual concepts served in lieu of real ABox instances.

5.2. Stable and Non-Stable Classifiers

Consider the following scenario: A rather shady-looking character produces from his capacious overcoat a large black box, which he claims is a seventh-generation classifier of terminological knowledge, guaranteed to produce sound (although not necessarily complete) inferences very quickly. We decide to test out his BBC (black-box classifier). First we store into the BBC the definitions of two concepts which we call A and B. We than ask the BBC "Does B specialize A?", and it responds (very quickly) "No". Next, we enter a few more definitions into the BBC, and again ask the BBC "Does B specialize A?" This time, its rapid rejoinder is "Yes"! Should we buy his BBC (this price is very reasonable)?

The answer is no: Let us define a *stable* classifier (or recognizer) to be one which produces the same answers to subsumption questions independently of additions or subtractions to/from the knowledge base (here we assume that no concept definitions are *modified*, and that at no time does the knowledge base contain undefined references). "Stability" is a highly-desirable feature in a TBox, because it provides a certain guarantee that when TBox knowledge is shared across several knowledge bases (e.g., by several applications) it will retain the same "meaning" in each of those contexts. We propose that "stability" be considered a test which serves to exclude some reasoners from being considered TBox classifiers. ("Soundness" should be another TBox requirement). The Loom Tbox of an example of a stable classifier; our friend's BBC is not stable.

The Loom UBox classifier/recognizer is not stable! Consider the Cars KB in Figure A-1. Suppose we make the following assertions

```
(assert (Motor-Vehicle BPV) (2-Person-Vehicle BPV)
        (Battery-Powered-Engine E)
        (has-component BPV E))
```

Now we ask, "Is BPV an instance of 2-Person-Car?" The UBox recognizer will make the following inferences

```
(Battery-Powered-Vehicle BPV)
(Car BPV)                    because of the "implies" axiom
(2-Person-Car BPV)
```

and conclude "Yes". However, if we remove the definition for `Battery-Powered-Vehicle` (or if it never existed) and re-run the UBox recognizer, it will not conclude either `(Car BPV)` or `(2-Person-Vehicle BPV)` .[8] On the other hand, if we run the Loom TBox recognizer on the same knowledge base and assertions, it will fail in both cases to recognize that BPV is a car (or a 2-person car). This behavior occurs because the axiom "`Battery-Powered-Vehicle` implies `Car`" is invisible to the TBox. The stability of the TBox classifier derives from the restrictions we place on what kinds of knowledge are classed as terminological in the TBox, not from the particular inference algorithm chosen -- we deliberately exclude from the TBox classes of knowledge which introduce non-stable behavior.

5.3. Modeling and Classification of Universal Knowle .ge

This section represents a long engineering note. We first describe the internal model adopted by Loom to represent universal knowledge, and then give some insight into the workings of Loom's UBox classification algorithm.

In a Loom concept network, separate objects, which we shall refer to as C_T and C_U, are defined to represent the TBox and UBox knowledge associated with a single concept C.[9] C_T contains exactly the definitional (terminological) component of C. C_U contains both the definitional *and* contingent knowledge knowledge associated with C. Thus, by construction, C_U always specializes C_T. An *implies* link links C_T to C_U, and has the meaning $\forall x[C_T(x) \rightarrow C_U(x)]$. In other words, C_T implies C_U.

Within a UBox concept, contingent restrictions and constraints are merged into a single definition, and are classified according to that definition. Suppose, for example, that we made the following declarations:

```
(defconcept C (:restriction R (:min 1))
   (:axioms (:restriction S (:min 1))))
(defconcept D (:restriction R (:min 1))
    (:restriction S (:min 1)))
```

[8] Note: This does *not* mean that it concludes "¬(Car BPV)". It merely *fails* to infer "(Car BPV)" -- the UBox classifier is not a non-monotonic reasoner.

[9] Browsers of Loom knowledge bases should be aware of the following: Loom maintains separate *name spaces* for TBox objects and UBox objects. In the TBox name space, only TBox objects are visible, and C_T has the name "C". In the UBox, only UBox objects are visible, and C_U has the name "C".

The classifier cannot distinguish between the objects C_U and D_T, and hence will merge these two concepts.

Implications are modeled as follows: Suppose we declare

```
(defconcept A ...
   (:axioms (:implies B)))
```

Rather than placing, say, an "implies" link between A_U and B_U, Loom captures the semantics of the implication axiom by merging all of the knowledge in B_U into A_U (in effect, "compiling out" the "implies" link). Equivalence relations add nothing new to the model, since they just consist of cycles of implication relations. If we declared that "A implies B", and also that "B implies A", Loom would merge B_U into A_U, and would also merge A_U into B_U, making A_U and B_U identical. The classifier would then merge these into a single UBox concept.

Loom's internal model of three of our original four categories of universal knowledge can thus be accomplished with the addition of only one new link, the "implies" link.[10] An important property of the model is that, in all cases, the "implies" links connect more general concepts to more specific ones: The Loom (and NIKL) TBox classifiers operate by picking an initial set of "starting points" (concepts) and then traversing down "subC" links which connect each concept to those concepts which directly specialize it. Loom's UBox classifier traverses down both "subC" and "implies" links. Because the "subC" and "implies" links form an acyclic directed graph, termination of the UBox classifier is guaranteed.

During the process of classifying/recognizing an object X in the UBox, the traversal of an "implies" link can cause knowledge to be acquired about X which is not entailed by its definition. This is the source of the "non-stability" in the UBox classifier. Recall the example in section 5.2 which traced the recognition of the object "BPV". One of the algorithm's starting points is the concept `2-Person-Vehicle`. If we visit its child `2-Person-Car` and make the test `(2-Person-Car X)` before having traversed the "implies" link between $\texttt{Battery-Powered-Vehicle}_T$ and $\texttt{Battery-Powered-Vehicle}_U$, we would receive a negative answer. Traversing that link causes us to acquire the knowledge `(Car BPV)`. After this point, the test `(2-Person-Car X)` returns in the affirmative. Hence, the first test to see if X was a 2-person car represented wasted effort.

One practical consequence of non-stability is that the ordering of subsumption tests is more critical for UBox classification than for TBox classification. Furthermore, it is not always the case that careful ordering of subsumption tests can avoid the necessity to repeat some subsumption tests (unless you have an "oracle" at your disposal). Theoretically, UBox classification could be significantly slower than TBox classification. We have not yet performed empirical tests which compare the relative performance of the two algorithms, but we expect that we will be able to achieve reasonable performance from the UBox.

6. Default Knowledge

Loom establishes a separate "box" for representing "default knowledge" -- knowledge representing statements that are "typically" true, but which are not axiomatic. Conceptually, this default knowledge consists of rules of the form "if nothing has been asserted or deduced which contradicts X, then assume X".

We will first discuss why the Loom architecture includes a Default Box. Then we will examine the semantics of the default value and closed-world-assumption constructs. Finally, we will preview what the operation of a non-monotonic classifier might look like.

[10] The fourth category "other" is handled by special-purpose data structures and algorithms which are outside of the scope of this discussion

6.1. The Case for a Default Box

We reject the idea of combining assertional and default knowledge into a single "non-monotonic ABox". Such a strategy would contradict a philosophical goal of the Loom architecture: We wish to reserve the ABox for statements about *individuals*, and to extend the representational power of the non-ABox portion of the system so that all statements about "classes of individuals" can be represented somewhere else other than in the ABox. The nature of default knowledge is that it generally makes statements about classes of individuals. Thus, we must consider what the implications are of developing yet another box.

The prerequisites for defining a new "box" in the Loom knowledge representation framework are that (i) we can identify a significant body of knowledge which would be assigned to that box, and (ii) a specialized reasoning facility must exist to process the inferences associated with this knowledge. The Loom system does not yet meet these requirements, because it is able to respond to only two very specialized forms of default knowledge -- it includes a limited treatment of default values, and it recognizes certain closed-world assumptions. On the other hand, we already have some idea of what a (much more general) non-monotonic classifier would look like. Its behavior is sketched below, in section 6.3. Therefore, we anticipate that both prerequisites will be met in a future version of Loom.

6.2. Default Values and Closed-World Assumptions

A "default value" is a value which is assigned to fill a role/slot for some individual in the absence of any explicitly-asserted (or derived) knowledge about that role filler. For example, in our Engines and Cars KB, the form

```
( defaults ( restriction type-of-fuel
                         ( vr Gasoline)))
```

in the `defconcept` declaration for `Internal-Combustion-Engine` declares that Gasoline is the default value for the role `type-of-fuel`. If for some constant `"x"` we have asserted `(Internal-Combustion-Engine x)` , and we have made no assertions of the form `(type-of-fuel x f)` , then the default assumption is `(type-of-fuel x Gasoline)` .

The act of assigning a default value can trigger a re-classification of an ABox object. For example, after making the assertion `(assert Elephant E1)`, the process of classifiying `E1` as an elephant could trigger a default assertion `color E1 Grey`, which might then cause E1 to be re-classified as a grey-elephant (if such a concept existed). We have yet to investigate whether default values may trigger cycles of reclassifications, and, if so, how the semantics of assigning default values should be restricted to prevent such cycles.

The Loom representation of closed-world assumptions is another example where we can elicit useful default behavior in the absence of a general-purpose non-monotonic classifier. Each ABox knowledge base is assumed to have either a "closed-world" or an "open-world" interpretation. "Open-world" means that in addition to the assertions about an individual that are explicitly stated in the knowledge base, there may be other relevant assertions which have been left unstated. For example, consider the Engines and Cars KB once again. Suppose we make the assertions

```
(assert (Internal-Combustion-Engine e)
        (Cylinder c1) (Cylinder c2)
        (Cylinder c3) (Cylinder c4)
        (cylinder e c1) (cylinder e c2)
        (cylinder e c3) (cylinder e c4))
```

Can we deduce `(4-Cylinder-Engine e)` ? The answer is no if we adopt an open-world assumption, because the possibility exists that there are 4 (or 12, or whatever) more cylinders which are also components of the engine "e". On the other hand, adopting a closed-world assumption would allow us to conclude that the four

cylinders which are components of "e" are the only ones that exist, in which case the inference `(4-Cylinder-Engine e)` is valid.

Loom allows one to declare selective "regions" of closed-world semantics within an open-world knowledge base: The declaration

```
(defrelation M ...
   (:axioms (:domain D))
   (:defaults closed-world-assumption))
```

has the following interpretation: "If "D(x)" has been asserted (or can be deduced) for some x, then for all y, "M(x, y)" is true only if it has been explicitly asserted, or can be derived." The `defrelation` declaration for the relation `cylinder` in the Engines and Cars KB includes such a closed-world assumption. This assumption allows the classifier to count instances of the `cylinder` relation when attempting to recognize an object as an instance of the concept `4-Cylinder-Engine`.

6.3. Preview of a Non-Monotonic Classifier

A non-monotonic classifier has not yet been developed for the Loom architecture. We provide here a preview of what its behavior will be like if and when it is constructed, with the intention of stimulating the demand for such a reasoner. Our example provides an illustration of how a classic problem in non-monotonic reasoning can be modeled by the Loom language.

In the process of classifying/recognizing an object "x", a non-monotonic classifier will reference both explicitly declared knowledge and default knowledge about "x", and hence may deduce classifications which are based on default assumptions. As is the case with UBox classification, the classifier may acquire additional information about "x" in the midst of the classification process. The possibility arises that the "acquired" knowledge will contradict one (or more) of the default assumptions. In this case, the classifier must retract any classifications it has already made which were based on these non-valid assumptions.

Consider the Birds KB in Figure A-5. Suppose we have made the assertion

```
(assert (Penguin Tweety))
```

The classifier may first deduce `(Bird Tweety)`, then pick-up the attached default implication and assume `(Flying-Animal Tweety)`, and then deduce `(Flying-Bird Tweety)`. Next, it may deduce `(Non-Flying-Animal Tweety)` from the definition of `Penguin`, and then discover that `Flying-Animal` and `Non-Flying-Animal` are disjoint. At this point, it must retract the earlier deductions `(Flying-Animal Tweety)` and `(Flying-Bird Tweety)`.

7. Conclusion

The Loom language introduces new expressivity and some new and powerful forms of inference into the KL-ONE paradigm for knowledge representation. The most significant achievement is the formulation of the UBox, which allows universal knowledge to be defined and reasoned about independently of the terminological knowledge. The UBox solves a long-standing problem of how to represent necessary and sufficient conditions, and provides a way for a user to introduce cyclic references into a knowledge base without derailing the classifier.

Looking towards the future, we have described the behavior of a Default Box, indicating how a classifier might be extended to perform non-monotonic classifications. Collectively, our results suggest that we have taken another step in an ongoing evolution of knowledge representation systems, wherein increasing numbers of specialized forms of reasoning can be organized within a principled knowledge representation framework.

References

[Brachman and Schmolze 85] Brachman, R.J., and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, August 1985, 171-216.

[Brachman, Fikes, and Levesque 83] Ronald Brachman, Richard Fikes, and Hector Levesque, "KRYPTON: A Functional Approach to Knowledge Representation," *IEEE Computer*, September 1983.

[Kaczmarek 86] T. Kaczmarek, R. Bates, G. Robins, "Recent Developments in NIKL," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, August 1986.

[Mac Gregor 87] Robert Mac Gregor and Raymond Bates, The Loom Knowledge Representation Language, 1987. (in preparation)

[Moore 86] Johanna D. Moore, NIKL Workshop Summary, 1986.

[Moser 83] M.G. Moser, "An Overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding*, Bolt, Beranek, and Newman, Inc., Cambridge, MA, 1983. BBN Technical Report 5421.

[Robins 86] Gabriel Robins, The NIKL Manual, 1986.

[Schmolze and Lipkis 83] James Schmolze and Thomas Lipkis, "Classification in the KL-ONE Knowledge Representation System," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

[Vilain 84] Marc Vilain, *KL-TWO, A Hybrid Knowledge Representation System*, Bolt Beranak and Newman, Technical Report 5694, September 1984.

[Vilain 85] M. Vilain, "The Restricted Language Architecture of a Hybrid Representation System," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, CA, August 1985.

[von Luck 87] K. von Luck, B. Nebel, C. Peltason, A. Schmiedel, *The Anatomy of the BACK System*, Technische Universitat Berlin, Technical Report KIT Report 41, January 1987.

A. Knowledge Bases

Engines and Cars Knowledge Base

```
(defrelation has-component :primitive (:inverse-of component-of))
(defrelation component-of :primitive)

(defconcept Horse-Power :primitive)
(defrelation horse-power (:range Horse-Power))

(defconcept Fuel :primitive)
(defrelation type-of-fuel (:range Fuel))
(defconcept Gasoline :primitive (:specializes Fuel))
(defconcept Diesel-Oil :primitive (:specializes Fuel))

... Engines
(defconcept Engine :primitive
  ( axioms (:restriction type-of-fuel (:number 1))
    (:restriction horse-power (:number 1))))

(defconcept Cylinder :primitive)
(defrelation cylinders (:specializes has-component) (:range Cylinder)
  (:defaults :closed-world-assumption))

(defconcept Internal-Combustion-Engine (:specializes Engine)
        (:restriction cylinders (:min 1))
  ( defaults
    ( restriction type-of-fuel (:vr Gasoline))))
(defconcept 4-Cylinder-Engine (:specializes Engine)
        (:restriction cylinders (:number 4)))

... Diesel-Engines
(defconcept Glow-Plug :primitive)
(defrelation compression-ratio :primitive
  ( axioms (:domain Internal-Combustion-Engine) (:range Integer)))
(defconcept Diesel-Oil-Engine (:specializes Engine)
        ( restriction type-of-fuel (:vr Diesel-Oil))
  ( axioms
    ( implies Diesel-Engine)))
(defconcept Thing-With-Glow-Plugs
        (:restriction (:vrdiff has-component Glow-Plug) ( min 1))
  ( axioms
    ( implies Diesel-Engine)))
(defconcept Very-High-Compression-Engine
        ( constraint greater-than (compression-ratio) 15)
  ( axioms
    ( implies Diesel-Engine)))
(defconcept Diesel-Engine  primitive
        ( specializes Internal-Combustion-Engine Diesel-Oil-Engine
                    Thing-With-Glow-Plugs Very-High-Compression-Engine))
(defconcept Battery-Powered-Engine  primitive ( specializes Engine))

... Cars
(defconcept Vehicle  primitive)
(defconcept Motor-Vehicle ( specializes Vehicle)
      ( restriction ( vrdiff has-component Engine) ( number 1)))
(defconcept Battery-Powered-Vehicle ( specializes Motor-Vehicle)
      ( restriction ( vrdiff has-component Engine) ( vr Battery-Powered-Engine))
  ( axioms ( implies Car)))
(defrelation occupants  primitive ( range Human))
(defconcept 2-Person-Vehicle ( specializes Vehicle)
      ( restriction occupants ( max 2)))
(defconcept Car  primitive ( specializes Vehicle)
  ( axioms
    ( implies Motor-Vehicle)))
(defconcept 2-Person-Car ( specializes Car 2-Person-Vehicle))
```

Figure A-1: Engines and Cars

Numeric Comparison Knowledge Bases

```
;;; Numeric Comparison Predicates
(defrelation numeric-comparison :primitive (:specializes Compute-Relation)
  (:axioms
     (:domain Real-Number) (:range Real-Number)
     (:covering greater-or-equal less-or-equal)))
(defrelation greater-than :primitive (:specializes greater-or-equal not-equal)
  (:annotation
    (:membership-test (lambda (domain range) (> domain range)))))
(defrelation less-than :primitive (:specializes less-or-equal not-equal)
  (:annotation
    (:membership-test (lambda (domain range) (< domain range)))))
(defrelation equal :primitive (:specializes greater-or-equal less-or-equal)
  (:annotation
    (:membership-test (lambda (domain range) (eql domain range)))))
(defrelation not-equal :primitive (:specializes numeric-comparison)
  (:axioms
    (:disjoint-covering greater-than less-than)))
(defrelation greater-or-equal :primitive (:specializes numeric-comparison)
  (:axioms
    (:disjoint-covering equal greater-than)))
(defrelation less-or-equal :primitive (:specializes numeric-comparison)
  (:axioms
    (:disjoint-covering equal less-than)))
```

Figure A-2: Numeric Comparison

Sets and Intervals Knowledge Base

```
... Sex
(defconcept Animal  primitive)
(defset Sex ( values Male Female) ( partitions Animal))

... Navy Rankings
(defconcept Navy-Person  primitive)
(defconcept Military-Rank  primitive)
(defrelation Rank ( range Military-Rank))
(defrelation Naval-Rank  primitive ( specializes Rank)
   ( axioms ( domain Navy-Person) ( range Naval-Rank)))

(definterval Naval-Rank  primitive ( specializes Military-Rank)
  ( values Seaman-Recruit Seaman-Apprentice Seaman Petty-Officer-Third-Class
           Petty-Officer-Second-Class Petty-Officer-First-Class Chief-Petty-Officer
           Senior-Chief-Petty-Officer Master-Chief-Petty-Officer
           Ensign Lieutenant-Junior-Grade Lieutenant Lieutenant-Commander
           Commander Captain Commodore Rear-Admiral Vice-Admiral Admiral)
  ( partitions Navy-Person ( suffix nil)))
(defset Naval-Officer-Rank ( specializes Naval-Rank) ( values [Ensign  Admiral]))

... Numbers
(defconcept Real-Number  primitive
  ( annotation
    ( membership-test (lambda (self) (numberp self)))))

(definterval Integer  primitive ( specializes Real-Number)
  ( values [-INFINITY  INFINITY])
  ( annotation
    ( membership-test (lambda (self) (integerp self)))
    ( predecessor-fn (lambda (self) (1- self)))
    ( successor-fn (lambda (self) (1+ self)))))
(definterval Natural-Number ( specializes Integer) ( values [0  INFINITY]))
(definterval Positive-Integer ( specializes Integer) ( values [1  INFINITY]))
(definterval Non-Negative-Integer ( specializes Integer)
    ( values ( INFINITY  -1) [1  INFINITY]))
```

Figure A-3: Sets and Intervals

Familial Relations Knowledge Base

```
... Person
(defconcept Person :primitive)

... Familial Relations
(defrelation parent :primitive
  ( :axioms ( :domain Person) ( :range Person)))
(defrelation father ( :specializes parent) ( :range Male))
(defrelation grandparent ( :composition-of parent parent))
(defrelation grandfather ( :composition-of parent father))
(defrelation ancestor ( :closure-of parent))
(defrelation child ( :inverse-of parent))
(defrelation sibling ( :composition-of parent child) ( :specializes not-equal))
(defrelation brother ( :specializes sibling) ( :range male))
```

Figure A-4: Familial Relations

Birds Knowledge Base

```
(defconcept Animal :primitive
   ( :axioms ( :disjoint-covering Flying-Animal Non-Flying-Animal)))
(defconcept Flying-Animal :primitive ( :specializes Animal))
(defconcept Non-Flying-Animal :primitive ( :specializes Animal))

(defconcept Bird :primitive ( :specializes Animal)
   (:defaults ( :implies Flying-Animal)))
(defconcept Flying-Bird ( :specializes Bird Flying-Animal))
(defconcept Penguin :primitive ( :specializes Bird Non-Flying-Animal))
```

Figure A-5: Birds

B. Semantics of Term-Defining Constructs

Loom Expression, e	Semantics of e, $[\![e]\!]$
(defconcept (specializes C_1 C_2))	$\lambda x.\ [\![C_1]\!](x) \wedge [\![C_2]\!](x)$
(defconcept (restriction M (vr C)))	$\lambda x.\ \forall y ([\![M]\!](x, y) \rightarrow [\![C]\!](y))$
(defconcept (restriction M (min n)))	$\lambda x.\ \exists$ n *distinct* $y_i \wedge_i [\![M]\!](x, y_i)$
(defconcept (restriction M (max n)))	$\lambda x.\ \exists$ n+1 *distinct* $y_i \wedge_i [\![M]\!](x, y_i)$
(defconcept (constraint CR (R_1 R_2) (S_1 S_2)))	$\lambda x.\ \forall y,z ([\![R_1]\!] \circ [\![R_2]\!](x, y) \wedge [\![S_1]\!] \circ [\![S_2]\!](x, z)) \rightarrow [\![CR]\!](y, z))$
(defconcept (constraint CR (R_1 R_2) v))	$\lambda x.\ \forall y ([\![R_1]\!] \circ [\![R_2]\!](x, y) \rightarrow [\![CR_v]\!](y, v))$
(defrelation (specializes M_1 M_2))	$\lambda x,y\ [\![M_1]\!](x, y) \wedge [\![M_2]\!](x, y)$
(defrelation (range C))	$\lambda x,y\ [\![C]\!](y)$
(defrelation (inverse-of M))	$\lambda x,y\ [\![M]\!](y, x)$
(defrelation (closure-of M))	$\lambda x,y\ [\![M]\!]^+(x, y)$
(defrelation (composition-of M_1 M_2))	$\lambda x,y\ [\![M_1]\!] \circ [\![M_2]\!](x, y)$

END

9-87

DTIC